МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
„ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних занять з дисципліни
"СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ"
для студентів напрямків "Прикладна математика",
"Системний аналіз" та "Інформатика"

Затверджено
редакційно-видавничою
радою університету,
протокол № 3 від 03.12.2008

Харків НТУ „ХПІ” 2009

Методичні вказівки до лабораторних занять з дисципліни “Системи штучного інтелекту” для студентів напрямків “Прикладна математика”, “Системний аналіз” та “Інформатика” / уклад. Ю.І. Дорофєєв. – Харків: НТУ “ХПІ”, 2009. – 40 с.

Укладач Ю.І. Дорофєєв

Рецензент В.П. Северин

Кафедра системного аналізу і управління

ВСТУП

Для людини розумові здібності мають дуже важливе значення. Протягом тисяч років людина намагається зрозуміти, як вона думає, тобто збагнути, як їй вдається сприймати, розуміти, прогнозувати й управляти світом, який є набагато значнішим за своїми розмірами і набагато складнішим у порівнянні з людиною. Перед фахівцями в галузі вивчення і створення інтелектуальних систем постає ще більш відповідальне завдання – не тільки зрозуміти природу інтелекту, але й створити інтелектуальні сутності.

У теперішній час тематика штучного інтелекту охоплює величезний перелік наукових напрямків, починаючи з таких задач загального характеру, як розпізнавання образів, класифікація і кластеризація даних, і закінчуючи такими спеціальними задачами, як гра в шахи та доведення математичних теорем. У штучному інтелекті систематизуються й автоматизуються інтелектуальні задачі, і тому ця галузь стосується будь-якої сфери інтелектуальної діяльності людини.

Метою цих методичних вказівок є допомога студентам засвоїти на практиці основні методи розв'язання інтелектуальних задач аналізу і обробки даних, а також принципи побудови та функціонування інтелектуальних систем. У результаті виконання лабораторних робіт студенти повинні вивчити типи архітектури інтелектуальних систем, способи їхньої побудови, навчання й моделювання.

Оскільки зазначена галузь досліджень характеризується значною новизною, даний посібник описує лише основні типи інтелектуальних задач, для розв'язання яких доцільно використати системи штучного інтелекту.

Дані лабораторні роботи спрямовані на набуття студентами практичних навичок використання сучасних програмних засобів для побудови і моделювання інтелектуальних систем аналізу та обробки даних.

Лабораторна робота 1

Побудова інтелектуальної системи розпізнавання образів, яка реалізує принцип самонавчання

1.1. Мета роботи

Розробити алгоритм, інтерфейс, написати й налагодити програму, яка реалізує інтелектуальну систему розпізнавання образів і здатна самонавчатися в процесі діалогу з користувачем.

1.2. Інформаційний матеріал

Головним критерієм інтелектуального рівня комп'ютерних програм є їхня здатність до навчання.

Існує два основних способи навчання людини. Навчати людину можна, викладаючи їй абстрактні факти, наприклад, можна сформулювати дитині загальне правило: торкнешся до гарячого − обпечешся. Інший спосіб − учити на конкретних прикладах: торкнувшись до гарячої плити − обпечешся, сунувши руку в багаття − обпечешся і т.п. Який би метод не використовувався при навчанні, людина завжди поповнює свої знання, стикнувшись із чимсь новим і зустрічаючи протиріччя. Наприклад, якщо дитина ще не знає, що зіткнення з гарячою водою призведе до опіку, і сунула палець в окріп, то, одержавши опік, вона може дійти потрібного висновку. Тобто людина змушена змінити свою уяву про навколишній світ, якщо довідалась, що її висновок виявився помилковим. Така форма отримання знань може бути названа «навчанням зі зворотним зв'язком». Людина, завдяки механізму зворотного зв'язку й уже наявним знанням (життєвому досвіду), здобуває нові знання. Принцип навчання зі зворотним зв'язком показаний на рис. 1.1.

Аналогічно можна реалізувати процес навчання комп'ютерної програми. Для цього програма повинна мати підсистему зберігання фактів і механізм виконання логічних висновків. Основний принцип: система буде самонавчатися тільки в тому випадку, якщо вона стикнулася з новими фактами, які суперечать раніше відомим.

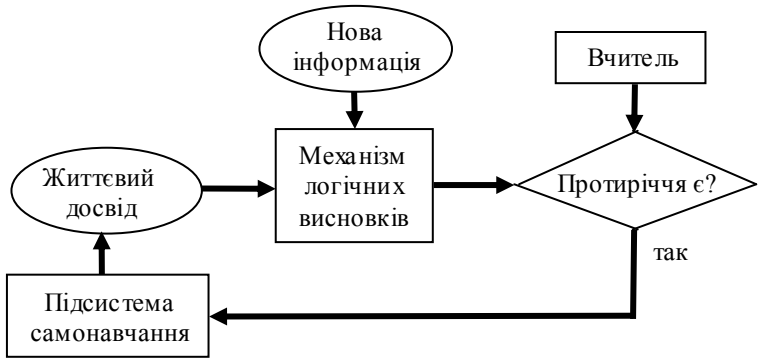

Рисунок 1.1 – Принцип навчання зі зворотним зв'язком

1.3. Постановка завдання

Розробити алгоритм, інтерфейс і написати програму, яка реалізує систему, призначену для розпізнавання двох об'єктів, які описані переліком ознак, а також реалізує принцип самонавчання. Програмне середовище вибрати самостійно.

Послідовність роботи програми:

1) Спочатку необхідно реалізувати режим початкового навчання системи. Для цього користувач у діалоговому режимі вводить назву «об'єкта 1» і назву «об'єкта 2».

2) Користувач указує ознаки, що характеризують «об'єкт 1» (кількість невідома). Система заносить їх у відповідний список.

3) Користувач указує ознаки, що характеризують «об'єкт 2». Система також заносить їх у відповідний список.

4) Система перевіряє, чи немає ознак, що належать одночасно двом об'єктам. Якщо такі є, вони видаляються зі своїх списків і містяться в загальний список. На цьому початкове навчання закінчено.

5) Система переходить у режим розпізнавання невідомих об’єктів та пропонує користувачеві ввести ознаки невідомого об'єкта.

6) Система на підставі наявної інформації намагається розпізнати невідомий об'єкт і повідомляє свій висновок користувачеві. Висновок повинен бути зроблений, навіть якщо наявної інформації недостатньо (наприклад, ознаки невідомого об'єкта ще не зустрічалися).

7) Після того як система повідомила свій висновок, вона запитує в користувача підтвердження чи спростування свого висновку.

8) Якщо висновок виявився правильним, самонавчання не відбувається, а списки ознак лише поповнюються (якщо зустрілися нові).

Якщо висновок виявився помилковим, система повинна шляхом самонавчання відкоригувати свою базу знань («список ознак об'єкта 1», «список ознак об'єкта 2» та «загальний список») так, щоб усунути наявне протиріччя.

9) Система запитує ознаки нового невідомого об'єкта і процес розпізнавання повторюється.

Контрольні запитання

1. За яких умов у системі реалізується принцип самонавчання?

2. Чи можна, використовуючи ті ж принципи, побудувати систему, здатну розпізнавати об'єкти, які належать до трьох класів?

Лабораторна робота 2

Синтез і навчання системи розпізнавання образів на основі процедури паралельної класифікації

2.1. Мета роботи

Розробити інтерфейс та побудувати інтелектуальну систему розпізнавання образів, яка працює на підставі процедури паралельної класифікації. Реалізувати настроювання параметрів розробленої системи за допомогою алгоритму «навчання зі вчителем».

2.2. Інформаційний матеріал

Розпізнавання (класифікацію) образів можна визначити як віднесення спостережуваних образів до певного класу на основі виділення та аналізу істотних ознак, які характеризують ці образи.

Припустимо, що об'єкт, який підлягає класифікації, описаний за допомогою вектора ознак $X = x_i$, $i = \overline{1, N}$, і може належати одному з C класів.

У системі класифікації, яка побудована на основі паралельної процедури, обчислюється множина функцій $y_j = f_j(x_1, x_2, ..., x_N),\ j = \overline{1, C}$, і об'єкт належить до класу j тоді й тільки тоді, коли $y_j = \max_k y_k,\ k = \overline{1, C}$. На практиці звичайно обчислюється зважена сума $y_j = \sum_{i=1}^{N} w_{ji} x_i$, де вектор $w_{ji},\ i = \overline{1, N}$ є вектором ваг j-го каналу. Якщо сума перевищує встановлений поріг w_o^j, об'єкт вважається належним до класу j.

Таким чином, система паралельної класифікації складається з C паралельних каналів, які відрізняються наборами вагових коефіцієнтів і порогових значень. Структура одного з каналів системи наведена на рис. 2.1.

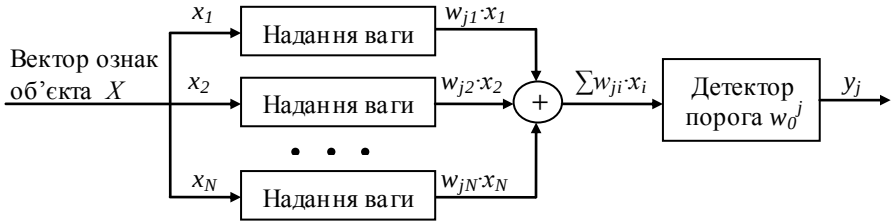

Рисунок 2.1 − Структура *j*-го каналу системи паралельної класифікації

У векторно-матричному вигляді зв'язок між входом і виходом системи паралельної класифікації можна подати таким чином:

$$Y = f(W \cdot X + W_0), \qquad (2.1)$$

де W − матриця вагових коефіцієнтів системи розмірністю $(C \times N)$; W_0 − вектор порогових значень системи розмірністю C; f − порогова функція.

Настроювання (навчання) системи паралельної класифікації здійснюється шляхом пред'явлення системі деякого набору об'єктів, що підлягають розпізнаванню, який називається навчальною вибіркою. Метою навчання є процес уточнення значень матриці ваг W і вектора порогів системи W_0.

Алгоритм «навчання з учителем» системи паралельної класифікації:

1) Сформувати навчальну вибірку. Кожен навчальний шаблон складається з двох векторів: вхідний вектор ознак об'єкта X і відповідний йому

бажаний вихідний вектор системи $\breve{Y}$. Образи, що належать різним класам, повинні чергуватися. Вхідні дані необхідно масштабувати так, щоб значення векторів належали інтервалу [0, 1]. Для цього знаходять максимальне значення кожної з ознак $x_i^{\max}$ і ділять всі значення на відповідні максимальні:

$$x_i^{норм} = \frac{x_i}{x_i^{\max}},\ i = \overline{1, N}, \tag{2.2}$$

де N – кількість ознак у вхідному векторі.

2) Присвоїти матриці вагових коефіцієнтів системи W і вектору порогів W_0 випадкові значення, розподілені в інтервалі [−0,3 0,3].

3) Подати на вхід системи черговий навчальний шаблон. Обчислити реальний вихідний вектор системи Y за формулою (2.1) і відповідний вектор помилки за формулою:

$$E = \left\| \breve{Y} - Y \right\|, \tag{2.3}$$

де $\breve{Y}$ − ідеальний вихідний вектор.

4) Обчислити величини корекції вагових коефіцієнтів і порогових значень за формулами:

$$\Delta W = E \cdot X_{норм}^T, \quad \Delta W_0 = E\ . \tag{2.4}$$

5) Обчислити нові значення матриці вагових коефіцієнтів і вектора порогів за формулами:

$$W(k+1) = W(k) + \Delta W, \quad W_0(k+1) = W_0(k) + \Delta W_0, \tag{2.5}$$

де k − номер ітерації.

6) Перевірити критерій закінчення процесу навчання. Якщо він виконаний, завершити навчання. У протилежному випадку перейти до п.3.

2.3. Постановка завдання

Варіант 1

1) Синтезувати систему розпізнавання образів, призначену для класифікації спортсменів за двома ознаками:

а) футболісти – зріст більше 170 см, вага більше 65 кг;

б) жокеї – зріст не більше 170 см, вага не більше 65 кг;

в) атлети – зріст і вага не відповідають першим двом класам.

2) Виконати процес навчання системи, використовуючи наведений алгоритм «навчання з учителем».

Варіант 2

1) Синтезувати систему розпізнавання образів, призначену для класифікації автомобілів за двома ознаками:

а) вантажні – вантажопідйомність більше 500 кг,
кількість місць більше 4;

б) легкові – вантажопідйомність не більше 500 кг,
кількість пасажирських місць не більше 4;

в) інші – ознаки об'єкта не відповідають першим двом класам.

2) Виконати процес навчання системи, використовуючи наведений алгоритм «навчання з учителем».

Варіант 3

1) Синтезувати систему розпізнавання образів, призначену для класифікації комах за двома ознаками:

а) павуки – кількість кінцівок не менш 8, кількість крил менше 2;

б) жуки – кількість кінцівок менше 8, кількість крил не менш 2;

в) інші – ознаки об'єкта не відповідають першим двом класам.

2) Виконати процес навчання системи, використовуючи наведений алгоритм «навчання з учителем».

Варіант 4

1) Синтезувати систему розпізнавання образів, призначену для класифікації літературних творів за двома ознаками:

а) повісті – кількість діючих осіб не більше 5,
кількість сторінок не більше 100;

б) романи – кількість діючих осіб більше 5,
кількість сторінок більше 100;

в) інші – ознаки об'єкта не відповідають першим двом класам.

2) Виконати процес навчання системи, використовуючи наведений алгоритм «навчання з учителем».

Контрольні запитання

1. Визначити розмірність матриці вагових коефіцієнтів W для системи розпізнавання, яка здатна класифікувати 3-вимірні образи на 10 класів.

2. Які критерії можна використати для зупинення процесу навчання системи розпізнавання за допомогою алгоритму «навчання з учителем»?

Лабораторна робота 3

Синтез і навчання системи розпізнавання образів за допомогою алгоритму "навчання без учителя"

3.1. Мета роботи

Розробити інтерфейс та побудувати інтелектуальну систему розпізнавання образів, для настроювання параметрів якої використовується алгоритм «навчання без учителя».

3.2. Інформаційний матеріал

У випадках, коли приналежність об'єктів з навчаючої вибірки до певних класів невідома, для навчання системи розпізнавання застосовується «навчання без учителя».

Спочатку вибирається міра близькості або спосіб обчислення відстані між об'єктами $d(X_i, X_j)$. Потім задача розпізнавання розв'язується, наприклад, за допомогою послідовного алгоритму, який використовує поняття порога w_0. Відповідно до цього алгоритму випадковим образом вибирається точка X_1 в просторі ознак, що оголошується центром e_1 першого кластера. Наступна точка X_2 належить до першого кластера, якщо $d(X_2, e_1) \le w_0$. У протилежному випадку X_2 приймається за центр другого кластера $X_2 = e_2$ і т.д. На l-му кроці, коли вже є r кластерів, точка X_l або стає центром $(r+1)$-го кластера, або належить до того з кластерів, для якого $d(X_l, e_j) \le w_0$. Якщо таких кластерів декілька, вибирається той, до центра якого точка X_l ближче всього.

Найбільш популярним з алгоритмів розглянутого класу є метод "k-середніх". Нехай наявну сукупність образів $X_1, X_2, \ldots, X_n$ потрібно розбити на задане число k $(k \le n)$ однорідних кластерів.

Алгоритм "k-середніх", який реалізує «навчання без учителя»:

Крок 0. Вибрати міру близькості $d(X_i, X_j)$, що дозволяє оцінити відстань між образами X_i та X_j. Вибрати як еталони перші k образів наявної сукупності: $e_i^{(0)} = X_i$, $i = 1,\ldots,k$ та встановити їх ваги рівними одиниці $w_i^{(0)} = 1$, $i = 1,\ldots,k$.

Крок 1. Вибрати наступний образ X_{k+1} і виконати перерахування еталонів та їхніх ваг за правилом:

$$e_i^{(v)} = \begin{cases} \dfrac{w_i^{(v-1)} e_i^{(v-1)} + X_{k+v}}{w_i^{(v-1)} + 1}, & d(X_{k+v}, e_i^{(v-1)}) = \min\limits_{1 \le j \le k} d(X_{k+v}, e_j^{(v-1)}); \\ e_i^{(v-1)}, & в\_протилежному\_випадку \end{cases}$$

$$w_i^{(v)} = \begin{cases} w_i^{(v-1)} + 1, & d(X_{k+v}, e_i^{(v-1)}) = \min\limits_{1 \le j \le k} d(X_{k+v}, e_j^{(v-1)}) \\ w_i^{(v-1)}, & в\_протилежному\_випадку \end{cases}, \qquad i = 1,...,k,$$

де v – номер ітерації.

Якщо виявлено декілька (по i) однакових мінімальних значень $d(X_{k+v}, e_i^{(v-1)})$, то поточний образ X_{k+v} відносять до еталона з мінімальним номером i.

Крок 2. Якщо не досягнута стадія стійких (по v) значень еталонів, вибрати наступний образ і повторити дії кроку 1. Якщо наявна сукупність образів вичерпана, знову вибрати образ X_1, потім X_2 і т.д.

Після закінчення процесу навчання класифікація нового образу здійснюється відповідно до правила мінімальної відстані щодо еталонів, отриманих на останньому кроці алгоритму.

3.3. Постановка завдання

1) Синтезувати систему розпізнавання образів на основі послідовної процедури, призначену для класифікації студентів Вашої академічної групи на декілька класів за результатами модульного контролю. Кількість оцінок кожного студента повинна бути не менш 4. Кількість класів визначає викладач.

2) Здійснити навчання системи, використовуючи наведений алгоритм «навчання без учителя».

3) Перевірити працездатність системи на тестових прикладах.

Контрольні запитання

1. Для чого кожному еталону e_i приписується вага w_i, значення якої збільшується на одиницю при кожнім перерахуванні координат даного еталона?

2. Які критерії можна використати для зупинення процесу навчання системи розпізнавання за допомогою алгоритму «навчання без учителя»?

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ
ПРИЙНЯТТЯ ОПТИМАЛЬНИХ РІШЕНЬ В ІГРАХ

Лабораторна робота 4

Розробка виграшної стратегії для гри «хрестики-нулики»

4.1. Мета роботи

Розробити інтерфейс, алгоритм та реалізувати програму, яка моделює гру «хрестики-нулики» на квадраті 3×3 та гарантує вибір виграшної стратегії для гравця.

4.2. Інформаційний матеріал

Розглянемо клас ігор двох осіб з повною інформацією. У таких іграх беруть участь два гравці, які по черзі роблять свої ходи. У будь-який момент гри кожному гравцеві відомо все, що відбулося у грі до цього моменту і що може бути зроблено в даний момент. Гра закінчується або виграшем одного гравця (і програшем іншого), або нічиєю. До даного класу належать такі ігри, як шахи, шашки, Нім, «хрестики-нулики» та ін.

При розгляді ігрових програм гравцем вважається комп'ютер, а супротивником − людина. Розглянемо завдання пошуку виграшної стратегії для гравця, відправляючись від деякої фіксованої позиції гри (необов'язково початкової). Для формалізації і вивчення ігрових стратегій у класі ігор з повною інформацією може бути використаний підхід, оснований на редукції задач. Послідовне застосування для вихідної позиції гри схеми зведення ігрових задач до сукупності підзадач породжує І/АБО-дерево (І/АБО-граф), що називають деревом (графом) гри. Дуги ігрового дерева відповідають ходам гравців, вершини − позиціям гри, причому листи дерева − це позиції, у яких гра завершується виграшем, програшем або нічиєю. Зазначимо, що для конфігурацій, де хід належить гравцеві (комп'ютеру), в ігровому дереві виходить АБО-вершина, а для позицій, у яких ходить супротивник (людина), − І-вершина.

Метою побудови ігрового дерева є одержання вирішального дерева. Вирішальне дерево закінчується на позиціях, виграшних для гравця, і містить повну стратегію досягнення їм виграшу: для кожного можливого продовження гри, обраного супротивником, у дереві або графі є відповідний хід, що приводить до перемоги.

У більшості ігор побудувати повні вирішальні дерева (і знайти повні ігрові стратегії) не виявляється можливим. Розглянемо, наприклад, гру «хрестики-нулики» на квадраті 3×3. Гравець ходить першим і ставить хрестики, а супротивник − нулики. Гра закінчується, коли складений або рядок, або стовпець, або діагональ із хрестиків (виграє гравець) чи нуликів (виграє супротивник). Оцінимо розмір повного дерева гри: початкова вершина має 9 дочірніх вершин, кожна з яких у свою чергу − 8 дочірніх; кожна вершина глибини 2 має 7 дочірніх і т.д. Таким чином, число кінцевих вершин у дереві гри дорівнює 9!=362880, хоча багато шляхів у цьому дереві обриваються раніше на заключних вершинах.

У випадках, коли повний перегляд дерева гри неможливий, застосовують наступний підхід:

1) задають граничну глибину пошуку;

2) оцінюють перспективність позицій гри за допомогою евристичних функцій. Для цього в кожній позиції гри формується дерево можливих продовжень гри, що має задану глибину, і за допомогою деякої функції обчислюються оцінки кінцевих вершин такого дерева. Потім отримані оцінки поширюються нагору по дереву, і коренева вершина, що відповідає поточній позиції, одержує оцінку, яка дозволяє з'ясувати силу гравця в даній позиції.

Мінімаксна процедура

Нижче пропонується один з варіантів обчислення оціночної функції. Кожній кінцевій позиції присвоюється одне з чотирьох значень ваги. Якщо вага дорівнює 4, це значить, що гравець у цій позиції виграє. Якщо вага дорівнює 3, то з поточної позиції не ясно, хто з гравців виграє зрештою. Вага, яка дорівнює 2, означає, що позиція приводить до нічиї. І нарешті, вага, яка дорівнює 1, означає, що виграє супротивник.

Розрізняють два види оцінок: статичні і динамічні. *Статичні оцінки* приписуються кінцевим вершинам дерева пошуку та обчислюються на основі евристичних правил. *Динамічні оцінки* випливають при поширенні статичних оцінок нагору по дереву. Відповідно до *мінімаксного* методу це робиться наступним чином. Для вершини, у якій хід виконує гравець (MAX), вибирається найбільша з оцінок вершин нижнього рівня (рівня MIN). У цьому випадку гравець зробить для себе найбільш вигідний хід. Для вершини, у якій хід виконує супротивник (MIN), вибирається найменша з

оцінок дочірніх вершин, тому що супротивник прагне зробити хід, який є найменш вигідним гравцеві. Таким чином, гравець MAX повинен віддати перевагу ходу, що веде в стан з максимальним значенням, а супротивник MIN − ходу, що веде у стан з мінімальним значенням.

На рис. 4.1 показаний приклад поширення статичних оцінок нагору по пошуковому дереву відповідно до зазначеного мінімаксного методу. З рисунка видно, наприклад, що кращим ходом гравця MAX у вершині *A* буде хід *A-B*, а кращим ходом супротивника MIN у вершині *D* буде хід *D-E*.

Якщо позначити статичну оцінку у вершині *P* через *h(P)*, а динамічну – через *H(P)*, то формально оцінки, які приписуються вершинам відповідно до мінімаксного методу, можна записати таким чином:

якщо *P* – кінцева вершина дерева пошуку:

$$H(P) = h(P);$$

якщо *P* – вершина, що відповідає ходу гравця MAX:

$$H(P) = \max_i H(P_i);$$

якщо *P* – вершина, що відповідає ходу супротивника MIN:

$$H(P) = \min_i H(P_i),$$

де P_i – дочірні вершини для вершини *P*.

Щоб програмно реалізувати мінімаксний метод визначення виграшної стратегії гравця, необхідно створити підпрограму, що обчислює оцінку позиції, перевіряючи всі можливі ходи. Для цього вона повинна рекурсівно викликати себе доти, поки не відбудеться одна з трьох подій. По-перше, вона може дійти до позиції, у якій гравець виграє. У цьому випадку вона присвоює позиції оцінку 4, що вказує на виграш гравця, який здійснив останній хід (якщо виграє супротивник, позиції присвоюється оцінка 1). По-друге, підпрограма може знайти позицію, у якій жоден з гравців не може зробити наступний хід. Гра при цьому закінчується нічиєю, тому позиції присвоюється оцінка 2. І нарешті, підпрограма може досягти заданої максимальної глибини рекурсії. У цьому випадку позиції присвоюється оцінка 3, що вказує на неможливість визначити переможця. Потім оцінки поширюються нагору по дереву пошуку, поки не буде присвоєна оцінка поточній позиції. Після цього гравець вибирає для наступного ходу позицію з максимальною оцінкою.

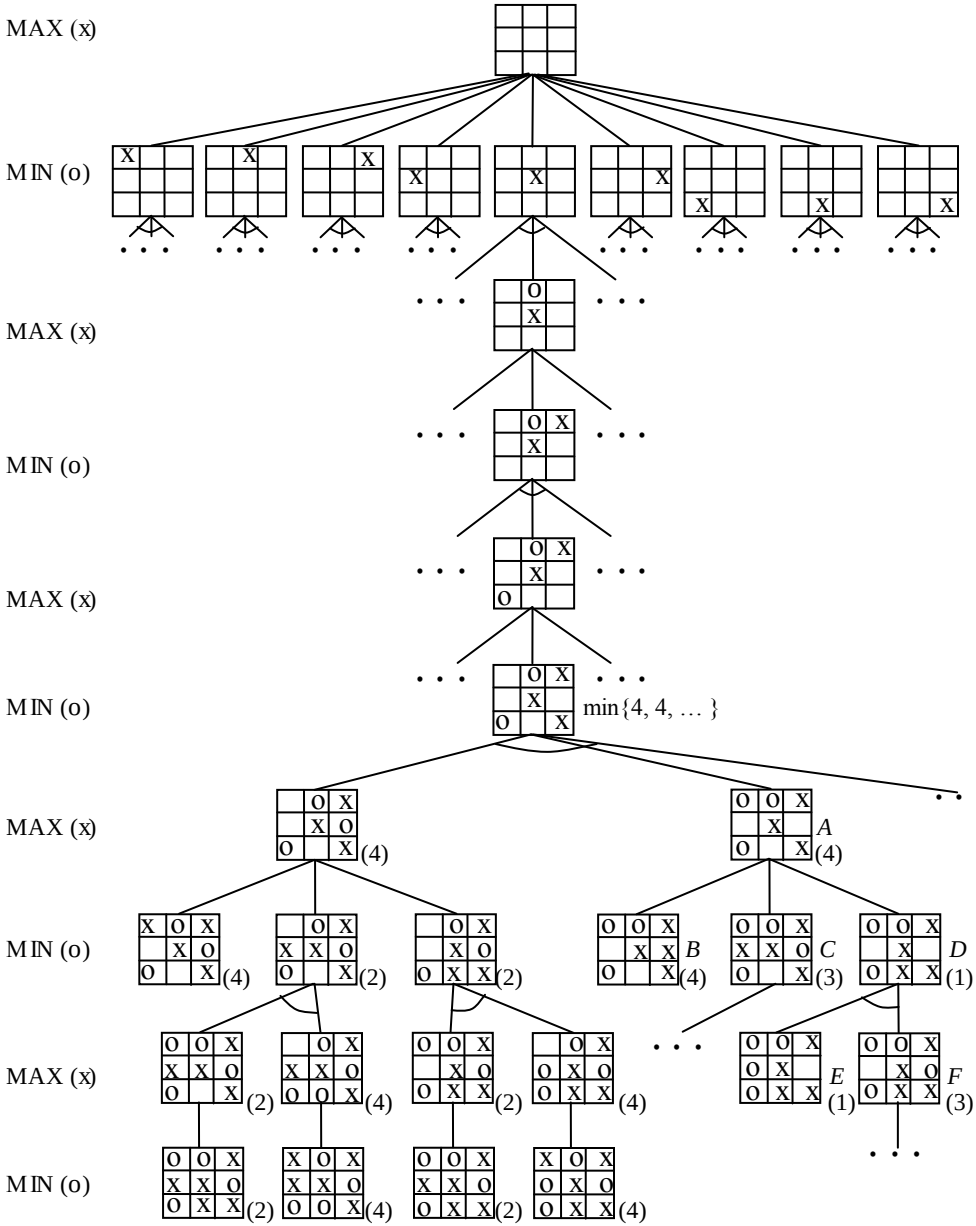

Рисунок 4.1 – І/АБО дерево пошуку (часткове) для гри «хрестики-нулики»

4.3. Постановка завдання

1) Розробити алгоритм гри «хрестики-нулики» на квадраті 3×3, що реалізує виграшну стратегію для "гравця MAX", який виконує перший хід.

2) Написати й налагодити програму гри, у якій роль "гравця MAX" грає комп'ютер, а роль "супротивника MIN" надається користувачеві.

3) Програмне середовище та інтерфейс вибрати самостійно.

Контрольні запитання

1. Для гри «хрестики-нулики» запропонуйте свій спосіб обчислення статичних оцінок і поясніть принцип мінімаксної процедури вибору ходу.

2. Що являє собою рішення при пошуку на І/АБО графі?

Лабораторна робота 5

Розробка виграшної стратегії для гри Нім

5.1. Мета роботи

Розробити інтерфейс, алгоритм та реалізувати програму, яка моделює гру Нім і гарантує вибір виграшної стратегії для гравця.

5.2. Інформаційний матеріал

Розглянемо клас ігор двох осіб з повною інформацією. У таких іграх беруть участь два гравці, які по черзі роблять свої ходи. У будь-який момент гри кожному гравцеві відомо все, що відбулося у грі до цього моменту і що може бути зроблено в даний момент. Гра закінчується або виграшем одного гравця (і програшем іншого), або нічиєю. До даного класу належать такі ігри, як шахи, шашки, Нім, «хрестики-нулики» та інші.

Розглянемо докладніше популярну гру Нім, що з'явилася в Китаї. Назву їй дав професор математики Гарвардського університету Чарльз Л. Бутон, який в 1901 році вперше опублікував повний аналіз гри. Відомі різні варіанти гри Нім. Зупинимося на класичному варіанті. Перед гравцями розташовується поле з каменями. Правила гри такі:

- камені розкладаються в кілька купок;
- гравці по черзі забирають камені з купок;
- не дозволяється за один хід брати камені з декількох купок;
- за один хід гравець повинен взяти хоча б один камінь;

• виграє той, хто візьме останній камінь.

Відкрита Ч. Бутоном [1] оптимальна стратегія гри основана на двійковій системі числення. Кожну позицію він визначив як небезпечну або безпечну. Якщо позиція, що створилася після чергового ходу гравця, гарантує йому перемогу, вона називається безпечною. У протилежному випадку - позиція небезпечна.

Ч. Бутон упевнено довів, що будь-яку небезпечну позицію завжди можна перетворити в безпечну за допомогою відповідного ходу. З іншого боку, якщо перед черговим ходом гравця вже склалася безпечна позиція, то будь-який його хід перетворює позицію в небезпечну. Таким чином, оптимальна стратегія гравця полягає в тому, щоб кожним ходом небезпечну позицію перетворювати в безпечну й змушувати суперника перетворювати позицію в небезпечну. Використання оптимальної стратегії гарантує перемогу гравця тоді і тільки тоді, коли він ходить першим і початкова позиція є небезпечною або він ходить другим, а початкова позиція безпечна.

Для того щоб визначити, якою є позиція, потрібно кількість каменів у кожній купці записати у двійковій системі числення і обчислити суму чисел у кожному стовпці (розряді). Якщо ця сума парна, то позиція безпечна. Якщо сума хоча б в одному розряді непарна, то позиція небезпечна. Більш простий спосіб оцінки позиції полягає в тому, щоб подати кількість каменів у кожній купці у вигляді суми ступенів двійки, а потім викреслити всі пари однакових ступенів і підсумувати ступені, що залишилися. У результаті отримують так звану «нім-суму» для даної позиції. Це число називають також «числом Ганді» або «числом Спрега-Ганді» на честь Р. Спрега і П. Ганді, які незалежно один від одного розробили загальну теорію такого роду ігор, основану на числових оцінках кожної ігрової позиції.

Припустимо, наприклад, що на початку гри є три купки − з трьох, п'яти і семи каменів. Запишемо ці числа в такому вигляді:

3 = 2 + 1;

5 = 4 + 1;

7 = 4 + 2 + 1.

Викреслимо відповідні пари четвірок, двійок і одиниць. Сума того, що залишилося, дорівнює одиниці − це і є нім-сума для даної позиції. Позиція є безпечною в тому і тільки в тому випадку, якщо нім-сума для неї дорівнює

нулю. У протилежному випадку позиція є небезпечною (як у наведеному прикладі).

Для того щоб забезпечити свій виграш, гравцеві треба перетворити небезпечну позицію в безпечну. У даному прикладі, якщо взяти одну фішку з будь-якої купки, то нім-сума позиції дорівнюватиме нулю.

Існує зворотний варіант гри Нім, коли той гравець, що забирає останній камінь, вважається програвшим. У цьому випадку в стратегію необхідно внести зміни, що стосуються кінця партії. Для того щоб виграти, потрібно дотримуватися звичайної стратегії, причому таким чином, щоб залишити непарне число купок, які складаються з одного каменя.

Розглянемо, наприклад, гру Нім з однією купкою каменів і обмеженням максимальної кількості каменів, які гравець може взяти за один хід. Позначимо початкову кількість каменів N, а кількість, що гравець може взяти за один хід, k. Оцінимо розмір повного дерева гри: кожна вершина має k дочірніх вершин, при цьому глибина дерева визначається величиною відношення N/k. Тоді повне число вершин дерева гри дорівнює $\sum_{i=0}^{N/k} k^i$. Тому побудувати повне вирішальне дерево гри і знайти на ньому виграшну стратегію практично неможливо.

5.3. Постановка завдання

1) Розробити алгоритм гри Нім з однією купкою монет, що реалізує виграшну стратегію для гравця (комп'ютера).

При цьому супротивник (користувач) вибирає:

- варіант гри (пряма або зворотна);
- вихідну кількість монет N;
- максимальну кількість монет k, яку гравці можуть взяти за один раз,

а гравець (комп'ютер) визначає, кому надати перший хід.

2) Розробити інтерфейс гри, реалізувати і налагодити програму гри.

3) Програмне середовище вибрати самостійно.

Контрольні запитання

1. Сформулювати виграшну стратегію для гри Нім з однією купкою каменів.

2. Що потрібно змінити у стратегії, якщо обрано зворотний варіант гри?

Лабораторна робота № 6

Генерація асоціативних правил

6.1. Мета роботи

Ознайомитися з принципами побудови асоціативних правил та способами обчислення покажчиків, які характеризують корисність правил. Реалізувати програму, яка на основі наданого набору транзакцій генерує асоціативні правила з вказаними значеннями покажчиків.

6.2. Інформаційний матеріал

Однією з найпоширеніших задач аналізу даних є визначення наборів об'єктів, які часто зустрічаються, у великій множині наборів і наступна генерація асоціативних правил зі знайдених частих наборів.

Нехай є множина об'єктів $I = \{i_1, i_2, ..., i_j, ..., i_n\}$. Набори об'єктів T з множини I, які є доступними для аналізу, називаються транзакціями $D = \{T_1, T_2, ..., T_m\}$.

Поширеністю (підтримкою) набору T називається відношення кількості транзакцій, у які входить набір T, до загальної кількості транзакцій:

$$Supp(T) = \frac{|D_T|}{|D|},$$

де D – набір транзакцій, інформація про які доступна для аналізу; D_T – набір транзакцій, у які входить набір T.

Набір F називається частим, якщо значення його поширеності більше деякого мінімального значення, заданого користувачем $Supp(F) > Supp_{\min}$.

Асоціативні правила, які побудовані на основі набору F (тобто $X \cup Y = F$), є всіма можливими комбінаціями об'єктів, що входять у набір F, і мають такий вигляд:

«якщо X, то Y»,

де X – не є логічним вираженням (як у класифікаційних правилах), а є об'єктами з набору F, з якими зв'язані (асоційовані) об'єкти, що включені в частину Y правила. Наприклад, «*якщо {хліб, масло}, то {чай}*».

Оскільки асоціативне правило «*якщо X, то Y*» будується на підставі набору *F*, то воно має значення поширеності, яке дорівнює поширеності набору *F*:

$$Supp_{X \Rightarrow Y} = Supp_F = \frac{|D_{F=X \cup Y}|}{|D|}.$$

Значення поширеності (підтримки) правила показує, який відсоток транзакцій підтримує дане правило.

Для оцінки корисності асоціативних правил використовують, окрім поняття поширеності, також наступні поняття: достовірність правила, потужність правила.

Достовірністю правила «*якщо X, то Y*» називається відношення кількості транзакцій, у які входять набори *X* і *Y*, до кількості транзакцій, у які входить набір *X* :

$$Conf_{X \Rightarrow Y} = \frac{|D_{F=X \cup Y}|}{|D_X|} = \frac{Supp_{X \cup Y}}{Supp_X}.$$

Достовірність правила показує ймовірність того, що з наявності в транзакції набору *X* випливає наявність у ній набору *Y*. Тобто, якщо достовірність правила дорівнює 20 %, це означає, що при покупці товару *X* у кожному п'ятому випадку купують і товар *Y*.

Потужністю правила «*якщо X, то Y*» називається відношення достовірності правила до поширеності набору *Y*:

$$Impr_{X \Rightarrow Y} = \frac{|D_{F=X \cup Y}|}{|D_X| \cdot |D_Y|} = \frac{Supp_{X \cup Y}}{Supp_X \cdot Supp_Y}.$$

Чим більша потужність правила, тим сильніший вплив, який поява в транзакції набору *X* чинить на появу набору *Y*.

Поняття поширеності, достовірності та потужності використовуються в розв'язанні задачі пошуку асоціативних правил. Користувач задає мінімальні значення перерахованих величин і, нарешті, ті правила, які не задовольняють поставленим умовам, відкидаються і не включаються в рішення задачі.

Задача пошуку асоціативних правил зводиться до відшукування множини всіх частих наборів $L = \{F \mid Supp(F) > Supp_{\min}\}$ і генерації асоціативних правил зі знайденої множини *L*.

Для розв'язання задачі пропонується алгоритм Apriori [3], який був описаний в 1994 році. Алгоритм використовує одну із властивостей поширеності, яка стверджує: поширеність будь-якого набору об'єктів не може перевищувати мінімальної поширеності кожної з його підмножин: $Supp(F) \le Supp(E)$, при $E \subset F$.

Алгоритм працює поетапно. На i-му етапі визначаються всі i-елементні набори, які часто зустрічаються. Кожен етап складається з двох кроків:

1) формування кандидатів – створюється множина усіляких i-елементних наборів з об'єктів, які подані в множині транзакцій, що аналізується;

2) обчислюються значення поширеності наборів-кандидатів і відкидаються ті, поширеність яких менше мінімального значення, що визначив користувач.

Результатом роботи алгоритму є об'єднання всіх множин частих наборів L_i для всіх i.

Для підрахунку значень поширеності потрібно порівняти кожну транзакцію з кожним кандидатом. Алгоритм Apriori дозволяє скоротити обсяг обчислень, необхідних для відшукування частих наборів, використовуючи ефективний спосіб підрахунку, оснований на зберіганні кандидатів у хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з покажчиками на нащадків, а листи – на кандидатів.

6.3. Постановка завдання

Варіант 1

Вибрати програмне середовище і реалізувати алгоритм пошуку асоціативних правил на прикладі набору транзакцій, які подані в табл. 6.1.

Таблиця 6.1

№ з/п	Перелік придбаних товарів
1	{ хліб, молоко, шоколад, кава }
2	{ кава, шоколад }
3	{ пиво, чіпси }
4	{ хліб, ковбаса, пиво, чіпси }
5	{ ковбаса, молоко, хліб }
6	{ кока-кола, чіпси }
7	{ хліб, пиво, чіпси, ковбаса, молоко }
8	{ хліб, молоко, чіпси, пиво }

Результат надати у вигляді:

1) множини всіх частих наборів ($Supp_{\min} = 0{,}5$);

2) множини всіх можливих асоціативних правил, значення поширеності яких не менше 0,5; значення достовірності − не менше 0,75; значення потужності − не менше 1,5.

Варіант 2

Вибрати програмне середовище і реалізувати алгоритм пошуку асоціативних правил на прикладі набору транзакцій, які подані в табл. 6.2.

Таблиця 6.2

№ з/п	Перелік придбаних товарів
1	{ шоколад, кава }
2	{ кава, шоколад, пиріжок, сік }
3	{ чай, пиріжок }
4	{ кава, пиріжок }
5	{ пиріжок, чай }
6	{ сік, бутерброд }
7	{ пиріжок, шоколад, кава, чай }

Результат надати у вигляді:

1) множини всіх частих наборів ($Supp_{\min} = 0{,}4$);

2) множини всіх можливих асоціативних правил, значення поширеності яких не менше 0,75; значення достовірності − не менше 0,75; значення потужності − не менше 1,4.

Контрольні запитання

1. Чи можна наступні асоціативні правила, у яких збігаються набори X,

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_3\},$$

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_4\},$$

об'єднати в одне правило: $X = \{i_1, i_2\} \Rightarrow Y = \{i_3, i_4\}$?

2. Чи збігається значення достовірності асоціативних правил, які побудовані на підставі того самого набору, наприклад:

1) $X = \{i_1\} \Rightarrow Y = \{i_2\}$ і $X = \{i_2\} \Rightarrow Y = \{i_1\}$?

2) $X = \{i_1, i_2\} \Rightarrow Y = \{i_3\}$ і $X = \{i_1\} \Rightarrow Y = \{i_2, i_3\}$?

Лабораторна робота 7

Розв'язання задачі класифікації шляхом побудови класифікаційних правил

7.1. Мета роботи

Ознайомитися з принципами побудови класифікаційних правил для розв'язання задачі класифікації. Розробити програму, яка на підставі навчальної вибірки об'єктів генерує класифікаційні правила за допомогою алгоритму «1-RULE».

7.2. Інформаційний матеріал

Розглянемо формальну постановку задачі класифікації.

Нехай є множина об'єктів $I = \{i_1, i_2, ..., i_j, ..., i_n\}$. Кожен об'єкт описується набором змінних $i_j = \{x_1, x_2, ..., x_m, y\}$, де x_i – незалежні змінні, значення яких відомі і на підставі яких визначається значення залежної змінної y. При цьому множина значень залежної змінної y є кінцевою і визначає належність об'єкта до певного класа.

Задача класифікації полягає у визначенні значення залежної змінної досліджуваного об'єкта на підставі значень його незалежних змінних.

Для розв'язання задачі класифікації на підставі навчальної вибірки будується модель знаходження значення залежної змінної, котру називають функцією класифікації. Існує три способи побудови функції класифікації: класифікаційні правила, дерева рішень і математичні функції.

Класифікаційні правила складаються з двох частин – умови і висновку – і записуються у форматі:

ЯКЩО <умова>, ТО <висновок>.

Умовою правила є перевірка однієї або декількох незалежних змінних. Перевірки декількох змінних можуть бути об'єднані за допомогою логічних операцій "І", "АБО" та "НІ". Висновком є значення залежної змінної або розподіл її ймовірності по класах.

Основними перевагами класифікаційних правил є:

- легкість їхнього сприйняття людиною;
- запис природною мовою;
- відносна незалежність правил – в існуючий набір правил легко додати нове без необхідності змінювати вже існуючі.

Найпростішим алгоритмом формування правил для класифікації об'єктів є алгоритм «1-RULE», названий так тому, що він будує правила за значенням однієї незалежної змінної.

Для кожного можливого значення кожної незалежної змінної формується правило, що класифікує об'єкти з навчальної вибірки. При цьому у висновку правила вказується значення залежної змінної, котре найбільш часто зустрічається в об'єктів з обраним значенням незалежної змінної. Потім для кожного правила обчислюється величина помилки. Помилкою правила є відношення кількості об'єктів, що мають обране значення незалежної змінної, але не належать до обраного класу, до загальної кількості об'єктів навчальної вибірки, що мають обране значення незалежної змінної.

Таким чином, для кожного значення залежної змінної (кожного класу об'єктів) буде побудований набір правил. Після оцінки величини помилки правил кожного набору, для віднесення об'єкта до кожного із класів вибирається одне правило з найменшою помилкою.

Якщо яка-небудь незалежна змінна має речовинний тип, всю область значень такої змінної розбивають на інтервали. У результаті буде отриманий набір дискретних значень, з якими може працювати алгоритм «1-RULE».

На закінчення необхідно відзначити, що алгоритм «1-RULE», незважаючи на свою простоту, у багатьох випадках виявляється досить ефективним. Це пояснюється тим, що багато об'єктів дійсно можна класифікувати лише за однією ознакою.

7.3. Постановка завдання

Варіант 1

Вибрати програмне середовище і реалізувати алгоритм побудови правил класифікації «1-RULE» на прикладі задачі про проведення спортивних ігор за різних погодних умов. Як навчальна вибірка використовуються дані про погодні умови й рішення, проводити ігри чи ні, для 14 випадків, які подані в табл. 7.1.

Результат подати у вигляді двох класифікаційних правил (одне правило для кожного класу рішень) із вказівкою обчислених значень помилки для кожного правила.

Таблиця 7.1

№ з/п	Спостереження	Температура	Вологість	Наявність вітру	Рішення про проведення гри
1	Сонце	Жарко	Висока	Немає	Немає
2	Сонце	Жарко	Висока	Є	Немає
3	Хмарність	Жарко	Висока	Немає	Є
4	Дощ	Норма	Висока	Немає	Є
5	Дощ	Холодно	Норма	Немає	Є
6	Дощ	Холодно	Норма	Є	Немає
7	Хмарність	Холодно	Норма	Є	Є
8	Сонце	Норма	Висока	Немає	Немає
9	Сонце	Холодно	Норма	Немає	Є
10	Дощ	Норма	Норма	Немає	Є
11	Сонце	Норма	Норма	Є	Є
12	Хмарність	Норма	Висока	Є	Є
13	Хмарність	Жарко	Норма	Немає	Є
14	Дощ	Норма	Висока	Є	Немає

Таблиця 7.2

№ з/п	Вік	Приписання	Астігматизм	Ступінь зношування	Рекомендація
1	2	3	4	5	6
1	Юний	Короткозорість	Немає	Знижена	Немає
2	Юний	Короткозорість	Немає	Нормальна	М'які
3	Юний	Короткозорість	Є	Знижена	Немає
4	Юний	Короткозорість	Є	Нормальна	Тверді
5	Юний	Далекозорість	Немає	Знижена	Немає
6	Юний	Далекозорість	Немає	Нормальна	М'які
7	Юний	Далекозорість	Є	Знижена	Немає
8	Юний	Далекозорість	Є	Нормальна	Тверді
9	Літній	Короткозорість	Немає	Знижена	Немає
10	Літній	Короткозорість	Немає	Нормальна	М'які
11	Літній	Короткозорість	Є	Знижена	Немає
12	Літній	Короткозорість	Є	Нормальна	Тверді
13	Літній	Далекозорість	Немає	Знижена	Немає
14	Літній	Далекозорість	Немає	Нормальна	М'які
15	Літній	Далекозорість	Є	Знижена	Немає
16	Літній	Далекозорість	Є	Нормальна	Немає

Продовження таблиці 7.2

1	2	3	4	5	6
17	Старечий	Короткозорість	Немає	Знижена	Немає
18	Старечий	Короткозорість	Немає	Нормальна	Немає
19	Старечий	Короткозорість	Є	Знижена	Немає
20	Старечий	Короткозорість	Є	Нормальна	Тверді
21	Старечий	Далекозорість	Немає	Знижена	Немає
22	Старечий	Далекозорість	Немає	Нормальна	М'які
23	Старечий	Далекозорість	Є	Знижена	Немає
24	Старечий	Далекозорість	Є	Нормальна	Немає

Варіант 2

Вибрати програмне середовище і реалізувати алгоритм побудови правил класифікації «1-RULE» на прикладі задачі визначення типу контактних лінз. Як навчальна вибірка використовуються рекомендації щодо вибору типу лінз для 24 чоловік, які подані в табл. 7.2.

Результат подати у вигляді трьох класифікаційних правил (одне правило для кожного класу лінз) із вказівкою обчислених значень помилки для кожного правила.

Контрольні запитання

1. Наведіть геометричну інтерпретацію задачі класифікації об'єктів, для опису яких використовуються двовимірні вектори.

2. Назвіть переваги класифікаційних правил у порівнянні з іншими способами побудови функції класифікації.

Лабораторна робота 8

Розв’язання задачі класифікації шляхом побудови дерева рішень

8.1. Мета роботи

Ознайомитися з принципами побудови дерев рішень для розв’язання задачі класифікації. Розробити програму, яка на основі навчальної вибірки об’єктів будує дерево рішення за допомогою алгоритму ID3.

8.2. Інформаційний матеріал

Формальна постановка задачі класифікації наведена у розділі 7.2.

Дерево рішень − це спосіб зображення процедури класифікації в ієрархічній послідовній структурі. Звичайно кожен вузол дерева включає перевірку певної незалежної змінної. Кожному можливому значенню змінної відповідає гілка, що виходить із вузла дерева. Листи дерева рішень відповідають значенням залежної змінної, тобто певним класам.

Загальний принцип побудови дерев рішень полягає в рекурсивній розбивці множини об'єктів з навчальної вибірки на підмножини, що містять об'єкти, які належать до однакових класів. Особлива увага приділяється вибору змінної, за якою буде виконуватися розбивка. Загальне правило для вибору можна сформулювати в такий спосіб: обрана змінна повинна поділити множину об'єктів так, щоб отримані в підсумку підмножини складалися з об'єктів, які належать до одного класу, або були максимально наближені до цього. Різні алгоритми реалізують різні способи вибору незалежної змінної для розбивки.

Алгоритм ID3 (Iterative Dichotomizer 3) був запропонований Дж. Квінтаном у 1983 році. Він забезпечує індуктивне формування понять на основі прикладів, у результаті чого виконується побудова дерева рішень, яке охоплює всі приклади з навчальної вибірки.

Нехай I_k – це кількість об'єктів з множини I, що належать до класу y_k. Тоді ймовірність того, що випадково обраний об'єкт i_j із множини I буде належати класу y_k, дорівнює

$$p_k = \frac{|I_k|}{|I|}. \qquad (8.1)$$

Якщо розглядати навчальні приклади як несумісні повідомлення, ймовірності одержання яких обчислюються за формулою (8.1), тоді кількість інформації, вимірювана в бітах, для всієї сукупності повідомлень можна оцінити за допомогою міри американського математика К. Шеннона:

$$Info(I) = -\sum_{i=1}^{q} p_i \cdot \log_2 p_i , \qquad (8.2)$$

де q – кількість класів, до яких належать об'єкти навчальної вибірки.

Якщо на черговому кроці класифікації для розбивки вибирається ознака x_i, яка може набувати k значень $x_i = \{x_i^1, x_i^2, ..., x_i^k\}$, то множина I розбивається на k підмножин $\{I_1, I_2, ..., I_k\}$. Тоді очікуваний обсяг інформації, пов'язаний з побудовою піддерев, що відповідають підмножинам $\{I_1, I_2, ..., I_k\}$, буде дорівнювати:

$$Info_{x_i}(I) = \sum_{k=1}^{p} \frac{|I_k|}{|I|} \cdot Info(I_k), \qquad (8.3)$$

де $Info(I_k)$ – кількість інформації, що відповідає піддереву множини I_k.

Кількість інформації, обумовлена включенням тієї або іншої ознаки в дерево рішень, визначається як різниця між кількістю інформації, що відповідає всьому дереву рішень, і кількістю інформації відповідного піддерева, розташованого нижче вузла, обумовленого обраною ознакою класифікації. Тому кількість інформації, одержувана при виборі ознаки x_i, визначиться з виразу

$$G(x_i) = Info(I) - Info_{x_i}(I). \qquad (8.4)$$

На кожному кроці виконання алгоритму ID3 вибирається ознака x_i, яка забезпечує максимальну кількість інформації $G(x_i)$.

Процес рекурсивно триває доти, поки в кожному вузлі дерева рішень не залишаться об'єкти з одного класу.

Як приклад розглянемо задачу визначення ступеня ризику серцево-судинних захворювань за трьома ознаками. Навчальна вибірка містить дванадцять прикладів, які наведені у табл. 8.1.

Таблиця 8.1

№ з/п	Вік	Паління	Споживання жирів	Ступінь ризику
1	2	3	4	5
1	молодий	так	надлишок	високий
2	молодий	так	норма	середній
3	молодий	ні	надлишок	низький
4	молодий	ні	норма	низький
5	середній	так	надлишок	високий
6	середній	так	норма	середній
7	середній	ні	надлишок	середній

Продовження таблиці 8.1

1	2	3	4	5
8	середній	ні	норма	низький
9	літній	так	надлишок	високий
10	літній	так	норма	високий
11	літній	ні	надлишок	середній
12	літній	ні	норма	середній

За формулою (8.1) одержимо:

p(високий) = 4/12; p(середній) = 5/12; p(низький) = 3/12.

За формулою (8.2) одержимо:

$$Info(I) = -\frac{4}{12}\log_2(\frac{4}{12}) - \frac{5}{12}\log_2(\frac{5}{12}) - \frac{3}{12}\log_2(\frac{3}{12}) = 1{,}555 \text{ біт}.$$

За формулою (8.3) одержимо:

$$Info_{вік}(I) = \frac{4}{12}\left(-\frac{1}{4}\log_2(\frac{1}{4}) - \frac{1}{4}\log_2(\frac{1}{4}) - \frac{2}{4}\log_2(\frac{2}{4})\right) +$$

$$+\frac{4}{12}\left(-\frac{1}{4}\log_2(\frac{1}{4}) - \frac{2}{4}\log_2(\frac{2}{4}) - \frac{1}{4}\log_2(\frac{1}{4})\right) +$$

$$+\frac{4}{12}\left(-\frac{2}{4}\log_2(\frac{2}{4}) - \frac{2}{4}\log_2(\frac{2}{4}) - 0\right) = 1{,}333 \text{ біт};$$

$$Info_{паління}(I) = \frac{6}{12}\left(-\frac{4}{6}\log_2(\frac{4}{6}) - \frac{2}{6}\log_2(\frac{2}{6}) - 0\right) +$$

$$+\frac{6}{12}\left(0 - \frac{3}{6}\log_2(\frac{3}{6}) - \frac{3}{6}\log_2(\frac{3}{6})\right) = 0{,}959 \text{ біт};$$

$$Info_{споживання\_жирів}(I) = \frac{6}{12}\left(-\frac{3}{6}\log_2(\frac{3}{6}) - \frac{2}{6}\log_2(\frac{2}{6}) - \frac{1}{6}\log_2(\frac{1}{6})\right) +$$

$$+\frac{6}{12}\left(-\frac{1}{6}\log_2(\frac{1}{6}) - \frac{3}{6}\log_2(\frac{3}{6}) - \frac{2}{6}\log_2(\frac{2}{6})\right) = 1{,}459 \text{ біт}.$$

Нарешті, за формулою (8.4) одержимо:

$G(вік) = 1{,}555 - 1{,}333 = 0{,}222$ біт;

$G(паління) = 1{,}555 - 0{,}959 = 0{,}596$ біт;

$G(споживання\_жирів) = 1{,}555 - 1{,}459 = 0{,}096$ біт.

У такий спосіб на першому кроці для розбивки буде обрана незалежна змінна «*Паління*». Побудована частина дерева рішень буде виглядати так, як зображено на рис. 8.1.

Рисунок 8.1 - Розбивка дерева рішень на першій ітерації

8.3. Постановка завдання

Варіант 1

Вибрати програмне середовище і реалізувати алгоритм побудови дерева рішень ID3 на прикладі задачі визначення типу контактних лінз. Як навчальна вибірка використовуються рекомендації щодо вибору типу лінз для 24 чоловік, які подані в табл. 7.2.

Результат подати у вигляді дерева рішень із вказівкою обчислених значень кількості інформації для кожної незалежної змінної на кожному кроці алгоритму.

Варіант 2

Вибрати програмне середовище і реалізувати алгоритм побудови дерева рішень ID3 на прикладі задачі про проведення спортивних ігор за різних погодних умов. Як навчальна вибірка використовуються дані для 14 випадків, які подані в табл. 7.1.

Результат подати у вигляді дерева рішень із вказівкою обчислених значень кількості інформації для кожної незалежної змінної на кожному кроці алгоритму.

Контрольні запитання

1. Поясніть основну ідею алгоритму побудови дерева рішень ID3.
2. Сформулюйте правила вибору ознаки класифікації в алгоритмі ID3.

Лабораторна робота 9

Розв'язання задачі кластеризації даних за допомогою ієрархічних алгоритмів

9.1. Мета роботи

Ознайомитися з принципами роботи ієрархічних алгоритмів, які використовуються для розв'язання задачі кластеризації даних. Розробити програму, яка на основі навчальної вибірки будує повну дендрограму об'єктів за допомогою дівізимного чи агломеративного алгоритму.

9.2. Інформаційний матеріал

Розглянемо формальну постановку задачі кластеризації.

Нехай є множина об'єктів $I = \{i_1, i_2, ..., i_j, ..., i_n\}$. Кожен об'єкт описується набором змінних $i_j = \{x_1, x_2, ..., x_m\}$, значеннями яких є дійсні числа.

Задача кластеризації полягає в побудові множини

$$C = \{c_1, c_2, ..., c_k, ..., c_h\},$$

де c_k – кластер, що містить схожі один на одного об'єкти з множини I:

$$c_k = \{i_j, i_p \,|\, i_j \in I, i_p \in I, d(i_j, i_p) < \sigma\},$$

де $d(i_j, i_p)$ − міра близькості між об'єктами; σ − величина, що визначає міру близькості для включення об'єктів в один кластер.

Методи розбивки множини об'єктів на кластери можна поділити на ієрархічні та неієрархічні.

У неієрархічних алгоритмах число кластерів, а також умова зупинки звичайно визначається заздалегідь.

В ієрархічних алгоритмах фактично відмовляються від визначення числа кластерів і будують повне дерево вкладених кластерів − дендрограму. Даний термін підкреслює деревоподібну структуру діаграм.

Недоліки ієрархічних алгоритмів добре відомі: складність вибору оптимальної міри близькості об'єктів, негнучкість ієрархічних класифікацій. Проте зображення результатів кластеризації у вигляді дендрограми дозволяє одержати найбільш повне уявлення про структуру кластерів.

Ієрархічні алгоритми кластеризації розподіляються на дівізимні та агломеративні.

Дівізимні алгоритми на першому рівні подають усю множину об'єктів I

як єдиний кластер c_1. Потім на кожному рівні алгоритму один з існуючих кластерів ділиться на два дочірніх. Розподіл триває, поки всі кластери не стануть сиглетонами (тобто складаються з одного об'єкта).

Розглянемо докладніше один рівень роботи алгоритму.

У кластері, обраному для розподілу, визначається об'єкт, у якого середнє значення міри близькості від інших об'єктів у цьому кластері найбільше. Цей об'єкт видаляється із кластера c_1 і формує перший елемент другого кластера c_2. На кожному наступному кроці даного рівня об'єкт у кластері c_1, для якого різниця між середнім значенням міри близькості до об'єктів, що перебувають у c_2, і середнім значенням міри близькості до об'єктів, що залишаються в c_1, найбільша, переноситься в c_2. Перенос об'єктів із c_1 у c_2 триває доти, поки відповідні різниці не стануть негативними, тобто поки існують об'єкти, що знаходяться до об'єктів кластера c_2 ближче, чим до об'єктів кластера c_1.

Кожен наступний рівень алгоритму застосовує процедуру розподілу до одного із кластерів, отриманих на попередньому рівні. Кластер для розщеплення на кожному рівні можна вибирати, наприклад, з найбільшим діаметром, який обчислюється за формулою $D = \max_C d(i_p, i_q) \forall i_p, i_q \in C$.

В *агломеративних* алгоритмах на першому кроці вся множина об'єктів *I* подається як множина кластерів: $c_1=\{i_1\}, c_2=\{i_2\}, \ldots, c_n=\{i_n\}$.

На наступному кроці вибираються два найбільш близьких кластера (наприклад, c_p та c_q) і поєднуються в один загальний кластер. Нова множина буде складатися з $(n-1)$ кластерів: $c_1=\{i_1\}, c_2=\{i_2\}, \ldots, c_p=\{i_p,i_q\}, \ldots, c_n=\{i_n\}$.

Повторюючи процес, одержують послідовні множини кластерів, що складаються з $(n-2)$, $(n-3)$ і т.д. На n-му кроці вийде кластер, що складається з n об'єктів і збігається з первісною множиною *I*.

Якщо кластери p і q поєднуються в кластер r, то для обчислення відстані від нового кластера до кластера s використовується формула:

$$d(r,s) = \alpha_p d(p,s) + \alpha_q d(q,s) + \beta d(p,q) + \gamma \left| d(p,s) - d(q,s) \right|,$$

де $d(i_j, i_p)$ – міра близькості між об'єктами (або кластерами-сиглетонами), а значення коефіцієнтів $\alpha_p, \alpha_q, \beta, \gamma$ залежать від обраного методу

визначення відстані між кластерами. Наприклад, у методі найближчого сусіда (англ.: *Nearest neighbor*) такою відстанню вважається відстань між найближчими об'єктами різних кластерів і значення коефіцієнтів дорівнюють: $\alpha_p = 1/2,\ \alpha_q = 1/2,\ \beta = 0,\ \gamma = -1/2$.

Міру близькості між об'єктами $d(i_j, i_p)$ необхідно вибирати з урахуванням наявної інформації про характер даних, які описують об'єкти кластеризації.

У випадку, коли об'єкти, що підлягають кластеризації, можуть бути подані у вигляді точок *m*-мірного простору, як міра близькості може бути використана відстань між об'єктами:

1) Евклідова відстань: $d_2(x_i, x_j) = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}$;

2) відстань Хеммінга: $d_H(x_i, x_j) = \sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|$;

3) пікова відстань: $d_L(x_i, x_j) = \frac{1}{m} \sum_{k=1}^{m} \frac{\left| x_{ik} - x_{jk} \right|}{x_{ik} + x_{jk}}$.

Наприклад, пікова відстань припускає незалежність між змінними, що описують об'єкти, тобто є відстанню в ортогональному просторі.

9.3. Постановка завдання

У середині 30-х років XX сторіччя відомий статист Р.А. Фішер описав три класи ірисів (ці дані часто називають іриси Фішера). Кожен екземпляр описується чотирма параметрами: довжина і ширина чашелистника, довжина і ширина пелюстка. У табл. 9.1 подані дані по п'ятьох екземплярах для кожного класу ірисів.

Варіант 1

Вибрати програмне середовище і реалізувати дівізимний алгоритм кластеризації для множини об'єктів, які подані у табл. 9.1.

Результат кластеризації подати у вигляді дендрограми.

Варіант 2

Вибрати програмне середовище і реалізувати агломеративний алгоритм кластеризації для множини об'єктів, які подані у табл. 9.1.

Результат кластеризації подати у вигляді дендрограми.

Таблиця 9.1

№ з/п	Довжина чашелистника	Ширина чашелистника	Довжина пелюстка	Ширина пелюстка	Клас
1	5,1	3,5	1,4	0,2	Iris setosa
2	4,9	3,0	1,4	0,2	Iris setosa
3	4,7	3,2	1,3	0,2	Iris setosa
4	4,6	3,1	1,5	0,2	Iris setosa
5	5,0	3,6	1,4	0,2	Iris setosa
6	7,0	3,2	4,7	1,4	Iris versicolor
7	6,4	3,2	4,5	1,5	Iris versicolor
8	6,9	3,1	4,9	1,5	Iris versicolor
9	5,5	2,3	4,0	1,3	Iris versicolor
10	6,5	2,8	4,6	1,5	Iris versicolor
11	6,3	3,3	6,0	2,5	Iris virginica
12	5,8	2,7	5,1	1,9	Iris virginica
13	7,1	3,0	5,9	2,1	Iris virginica
14	6,3	2,9	5,6	1,8	Iris virginica
15	6,5	3,0	5,8	2,2	Iris virginica

Лабораторна робота 10

Оптимізація функції за допомогою генетичних алгоритмів

10.1. Мета роботи

Ознайомитися з принципами роботи та основними операторами генетичних алгоритмів. Розробити програму, яка розв'язує задачу оптимізації функції двох змінних за допомогою генетичного алгоритму.

10.2. Інформаційний матеріал

Генетичним алгоритмом є метод, що відображає природну еволюцію методів розв'язання інтелектуальних задач і в першу чергу − задач оптимізації. Генетичні алгоритми – це процедури пошуку рішення, основані на механізмах парної репродукції і успадкування, в яких використовується еволюційний принцип виживання найбільш пристосованих особин. Основні відмінності генетичних алгоритмів від традиційних методів оптимізації:

1) в процесі пошуку обробляються не значення параметрів задачі, а їх закодована форма;

2) пошук виконується не з однієї точки, а з декількох точок одночасно;

3) використовується лише цільова функція (функція пристосованості), а не її похідні або інша додаткова інформація;

4) застосовуються як ймовірнісні, так і детерміновані правила вибору.

Класичний генетичний алгоритм складається з таких кроків:

1) ініціалізація або вибір вихідної популяції хромосом;

2) оцінка пристосованості хромосом у популяції;

3) селекція хромосом;

4) створення нащадків обраних пар батьків і мутація нових особин;

5) формування нової популяції;

6) перевірка умови зупинки алгоритму і вибір «найкращої» хромосоми.

Ініціалізація полягає у випадковому виборі заданої кількості хромосом, які подаються двійковими послідовностями фіксованої довжини.

Оцінювання пристосованості хромосом у популяції полягає в розрахунку значення функції пристосованості для кожної хромосоми. Форма функції залежить від характеру розв'язуваної задачі. У задачах оптимізації функція пристосованості називається цільовою функцією, в теорії ігор − функцією вартості. Передбачається, що функція пристосованості завжди набуває додатних значень, і її потрібно максимізувати.

Селекція хромосом полягає у виборі пар хромосом, які братимуть участь у створенні нащадків. Такий вибір виконується відповідно до принципу природного відбору, згідно з яким найбільші шанси на участь у процесі розмноження мають хромосоми з найбільшими значеннями функції пристосованості. Таким чином, ймовірність відбору особини пропорційна значенню її функції пристосованості. В результаті процесу селекції створюється батьківська популяція, чисельність якої визначається розробником.

Формування нащадків здійснюється за допомогою оператора схрещування. Спочатку батьківські хромосоми випадковим чином об'єднуються в пари. Потім для кожної пари вибирається позиція гена в хромосомі, яка визначає точку схрещування. Якщо хромосома кожного з батьків складається з L генів, то визначення точки схрещування зводиться до випадкового вибору числа l_k з інтервалу $[1, L-1]$. Тоді в результаті схрещування з'являється нова пара нащадків:

1) нащадок, хромосома якого на позиціях від 1 до l_k складається з генів першого батька, а на позиціях від l_k+1 до L − з генів другого батька;

2) нащадок, хромосома якого на позиціях від 1 до l_k складається з генів другого батька, а на позиціях від $l_k + 1$ до L – з генів першого батька.

Оператор мутації із заданою ймовірністю p_m (зазвичай $0 \le p_m \le 0{,}1$) застосовується до породжених хромосом і змінює значення випадково вибраного гена в хромосомі на протилежне.

Хромосоми, отримані в результаті вживання генетичних операторів схрещування і мутації, включаються до складу нової популяції. Для них обчислюється значення функції пристосованості, і оператор редукції відновлює вихідний розмір популяції шляхом видалення хромосом з найменшими значеннями функції пристосованості.

Критерієм зупинки роботи генетичного алгоритму є одна з трьох подій:

1) сформовано задане користувачем число поколінь;

2) популяція досягла заданої користувачем якості, наприклад значення функції пристосованості всіх хромосом перевищило заданий поріг;

3) досягнутий деякий рівень збіжності, тобто хромосоми в популяції стали настільки подібними, що подальше їх поліпшення відбувається надзвичайно повільно.

Після завершення роботи алгоритму з кінцевої популяції вибирається хромосома, яка дає максимальне значення функції пристосованості і є, таким чином, результатом роботи генетичного алгоритму.

Як приклад, розглянемо задачу знаходження максимуму функції

$$f(x, y) = 2x^2 + \frac{3x}{y}, \qquad (10.1)$$

де x набуває дійсних значень з інтервалу $[0{,}0\ 1{,}5]$, а y набуває дійсних значень з інтервалу $[10{,}0\ 10{,}7]$. Припустимо, розв'язок необхідно знайти з точністю до одного знака після коми.

Пошук рішення зводиться до перегляду двовимірного простору, що складається з 16 точок по осі x: (0,0 0,1 … 1,5), і 8 точок по осі y: (10,0 ... 10,7). Значення x можна подати за допомогою бінарного ланцюжка, що складається з чотирьох елементів, оскільки за допомогою 4 біт можна отримати $2^4 = 16$ кодових комбінацій. Тоді ланцюжок [0000] відповідатиме числу 0,0, ланцюжок [0001] – числу 0,1 і так далі, ланцюжок [1111] – числу 1,5. Для подання значень y знадобиться 8 комбінацій: [000] відповідатиме чис-

лу 10,0, [001] – числу 10,1 і так далі, [111] – числу 10,7. У результаті кожна точка простору пошуку (x, y) буде подана бінарним ланцюжком з семи елементів: перші чотири кодують значення x, а останні три – значення y.

Ініціалізація. Нехай популяція складається з 8 хромосом. Необхідно випадковим чином згенерувати 8 двійкових послідовностей завдовжки 7 біт:
Популяція P(1): ch_1=[1110011] ch_3=[0111011] ch_5=[0100010] ch_7=[1010110]
ch_2=[0011001] ch_4=[0010001] ch_6=[1100110] ch_8=[0000101]

Оцінка функції пристосованості. Функція пристосованості в даному прикладі задається виразом (10.1). Обчислимо її значення для кожної хромосоми з вихідної популяції:

$F(ch_1) = 4,3$ $F(ch_3) = 1,2$ $F(ch_5) = 0,4$ $F(ch_7) = 2,3$
$F(ch_2) = 0,3$ $F(ch_4) = 0,1$ $F(ch_6) = 3,2$ $F(ch_8) = 0,0$

Селекція хромосом. Встановимо розмір батьківської популяції рівним 4. Виберемо з поточної популяції 4 хромосоми з найбільшими значеннями функції пристосованості і випадковим чином сформуємо з них 2 пари батьків: ch_1 і ch_6, ch_3 і ch_7.

Вживання операторів схрещування і мутації.

Перша пара батьків:

$$\left.\begin{array}{l} ch_1 = [1110011] \\ ch_6 = [1100110] \end{array}\right\} \text{ схрещування в точці } l_k = 3: \quad \begin{array}{l} ch_9 = [1110110] \\ ch_{10} = [1100011] \end{array}$$

Друга пара батьків:

$$\left.\begin{array}{l} ch_3 = [0111011] \\ ch_7 = [1010110] \end{array}\right\} \text{ схрещування в точці } l_k = 5: \quad \begin{array}{l} ch_{11} = [0111010] \\ ch_{12} = [1010111] \end{array}$$

$$ch_9 = [1110110] \quad \} \text{ мутація в точці } l_k = 4: \quad ch_9 = [1111110]$$

Формування нової популяції. Хромосоми, отримані в результаті вживання генетичних операторів до батьківської популяції, включаються до складу нової популяції. Потім обчислюються значення функції пристосованості для всіх хромосом і відновлюється вихідний розмір популяції шляхом відкидання хромосом з найменшими значеннями функції пристосованості:

$F(ch_1) = 4,3$ $F(ch_4) = 0,1$ $F(ch_7) = 2,3$ $F(ch_{10}) = 3,2$
$F(ch_2) = 0,3$ $F(ch_5) = 0,4$ $F(ch_8) = 0,0$ $F(ch_{11}) = 1,2$
$F(ch_3) = 1,2$ $F(ch_6) = 3,2$ $F(ch_9) = 4,9$ $F(ch_{12}) = 2,3$

Популяція P(2): ch_1=[1110011] ch_6=[1100110] ch_9=[1111110] ch_{11}=[0111010]
ch_3=[0111011] ch_7=[1010110] ch_{10}=[1100011] ch_{12}=[1010111]

Якщо умова зупинки роботи генетичного алгоритму виконана, то виводиться розв'язок задачі, яким є хромосома з найбільшим значенням функції пристосованості.

10.3. Постановка завдання

Варіант 1

За допомогою генетичного алгоритму знайти максимум функції

$$f(x, y) = \frac{3x^2}{y} + \frac{y^2}{5x},$$

де x набуває дійсних значень з інтервалу $[3{,}0\ 6{,}1]$, а y набуває дійсних значень з інтервалу $[5{,}0\ 6{,}5]$. Розв'язок необхідно знайти з точністю до одного знака після коми.

Варіант 2

За допомогою генетичного алгоритму знайти мінімум функції

$$f(x, y) = \frac{(x-1)^2}{y} + \frac{(y+1)^2}{x},$$

де x набуває дійсних значень з інтервалу $[3{,}0\ 4{,}5]$, а y набуває дійсних значень з інтервалу $[1{,}0\ 4{,}1]$. Розв'язок необхідно знайти з точністю до одного знака після коми.

СПИСОК ЛІТЕРАТУРИ

1. **Бондарев В.Н.** Искусственный интеллект : учеб. пособие для вузов / В.Н. Бондарев, Ф.Г. Аде. – Севастополь: Изд-во СевНТУ, 2002. – 615 с.

2. **Гаврилова Т.А., Хорошевский В.Ф.** Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб: Питер, 2000. – 384 с.

3. **Рассел С., Норвиг П.** Искусственный интеллект: современный подход, 2-е изд.: пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 1408 с.

Зміст

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних занять з дисципліни
“СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ”
для студентів напрямків “Прикладна математика”,
“Системний аналіз” та “Інформатика”

Укладач ДОРОФЄЄВ Юрій Іванович

Відповідальний за випуск О.С. Куценко

Роботу до видання рекомендував О.В. Горєлий

Редактор О.С. Самініна

План 2009 р., поз. 5/66-09

Підп. до друку 06.05.2009. Формат 60×84 1/16. Папір офсетний.
Друк – ризографія. Гарнітура Таймс New Roman. Ум. друк. арк. 2,3.
Наклад 50 прим. Зам. № 136. Ціна договірна.

Видавничий центр НТУ “ХПІ”.
Свідоцтво про державну реєстрацію ДК №116 від 10.07.2000 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21